Web Engineering

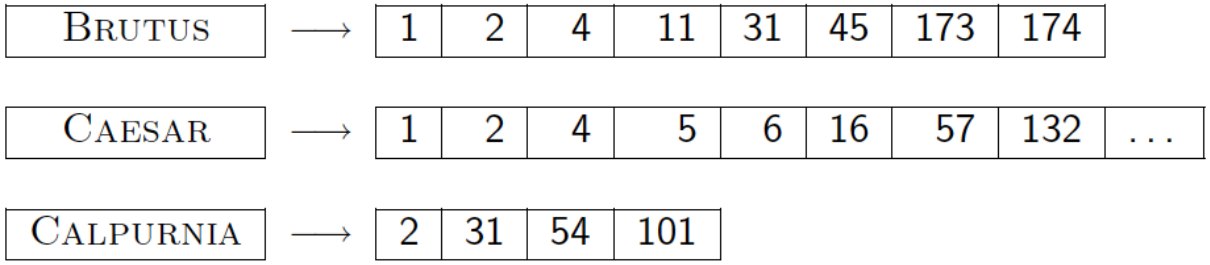
Lecture 4

Indexing - Part II

Building Term Vocabulary & Postings List

Recap of the previous lecture

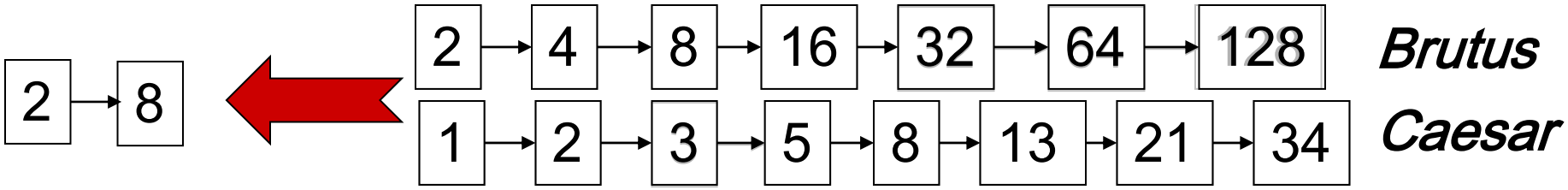
- Basic inverted indexes:
  - Structure: Dictionary and Postings



  - Key step in construction: Sorting
- Boolean query processing
  - Intersection by linear time “merging”
  - Simple optimizations

The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.

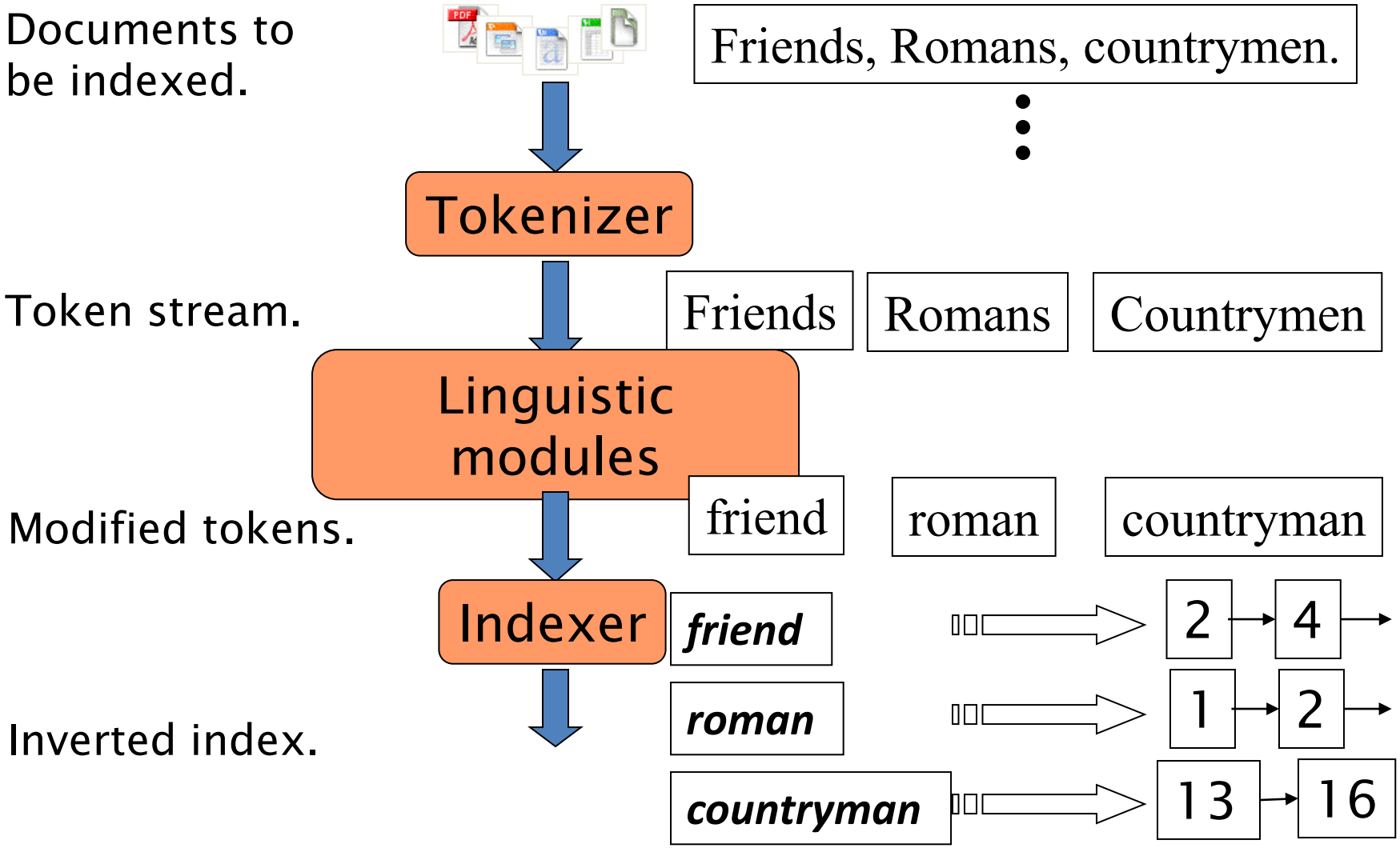
Crucial: postings sorted by docID.

Plan for the upcoming lectures

Elaborate basic indexing (this lecture)

- Preprocessing to form the term vocabulary
  - Documents
  - Tokenization
  - What *terms* do we put in the index?
- Postings (next lecture)
  - Faster merges: skip lists
  - Positional postings and phrase queries

Recall the basic indexing pipeline



Parsing a document

- Digital documents that are the input to an indexing process are typically **bytes** in a file.
- The first step of processing is to convert this ***byte sequence*** into a linear ***sequence of characters*** (***parsing***).
- To parse a document, the following properties of the document should be known:
  - The ***encoding mechanism*** of the text (ASCII/Unicode/vendor-specific standards) Then can decode the bytes to a character sequence.
    - For each language, each character is represented by certain number determined by the encoding used.
  - The ***document format***, and then an appropriate decoder has to be used.

Parsing: Language issues

- Some encodings makes reading mixed Arabic documents not linear; as characters are stored in memory in the same order of writing.
  - Arabic characters are stored from right to left and numbers are stored from left to right.
  - So, when reading this Arabic document, it has to reverse only Arabic digits but not numbers; making it not straightforward.

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← → ← → ← start

'Algeria achieved its independence in 1962 after 132 years of French occupation.'

- But ***Unicode*** stores numbers in the same direction of the surrounding text.
  - Thus, if Arabic precedes the number, then the number is stored on computer from right to left and the reverse operation occurs on the whole document, making it straightforward.

Choosing a Document Unit

- The next phase is to determine what the document unit for indexing is.
  - For example, we take each file in a folder as a document.
- What is a unit document?
  - A file?
  - An email? (Perhaps one of many in an mbox.)
  - An email with 5 attachments?
  - A group of files (PPT as HTML pages)

Choosing a Document Unit

- For a collection of books, it would usually be a bad idea to index an entire book as a document.
  - A search for Chinese toys might bring up a book that mentions China in the first chapter and toys in the last chapter, but this does not make it relevant to the query.
  - Instead, we may well wish to index each chapter or paragraph as a mini-document. Matches are then more likely to be relevant, and since the documents are smaller it will be much easier for the user to find the relevant passages in the document.

Choosing a Document Unit

- Indexing granularity affects IR performance:
  - If units are too large, difficult to get relevant information.
  - If units are too small, users can be overwhelmed by too many returns.

- For a good choice of granularity, the person who is deploying the IR system must have a good understanding of the document collection, the users, and usage patterns.

TOKENS AND TERMS

Tokenization

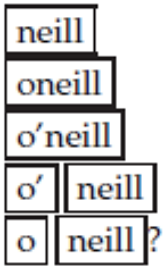
- Given a character sequence and a defined document unit, tokenization is the task of chopping it up into meaningful pieces, called ***tokens***.
  - Tokenization perhaps occur *at the same time* throwing away certain characters, such as punctuation (apostrophe, colon, comma, dash,!, semicolon, quotation mark).
- Input: "***Friends, Romans and Countrymen***"
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- Each such token is now a candidate for an index entry, after further processing.

Tokenization

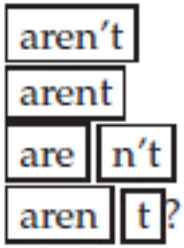
What are the correct tokens to use?

- ***What do you do about the various uses of the apostrophe for possession and contractions?***
  - *Example: Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.*

For *O'Neill*, which of the following is the desired tokenization?



And for *aren't*, is it:



- *A **simple strategy** is to just split on all non-alphanumeric characters.*
- *For either Boolean or free text queries, the **exact same tokenization of document and query words** must be done.*
  - *Generally, by processing queries with the same tokenizer.*
  - *This guarantees that a sequence of characters in a text will always match the same sequence typed in a query.*

Tokenization
What are the correct tokens to use?

Hyphenation is used for various purposes:

- ***Splitting up vowels in words (co-education)***
  - more commonly written as just one token (coeducation)
- ***Joining nouns as names (Hewlett-Packard)***
  - ***Hewlett*** and ***Packard*** as two tokens?
- ***To show word grouping (the hold-him-back and-drag-him-away maneuver)***
  - should be separated into words

- Handling hyphens automatically can thus be complex:
  - It is more commonly **done by** some heuristic **rules**, such as allowing short hyphenated prefixes on words, but not longer hyphenated forms.

Tokenization
What are the correct tokens to use?

- Computer technology has introduced ***new types*** of character sequences that a tokenizer should probably tokenize as a single token, including:
  - **Email addresses** (jblack@mail.yahoo.com),
  - **Web URLs** (http://stuff.big.com/new/specials.html),
  - **IP addresses** (142.32.48.231), etc.
- Older IR systems may not index numbers:
  - But often very useful: think about things like looking up error codes on the web.
- **Items that have a clear semantic type are often indexed separately as document "meta-data"**.
  - Creation date, format, etc.

Stop words

- **Stop words** are extremely common words which would appear to be of little value in helping select documents matching a user need.
  - Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in ,is, it, its, of, on, that, the, to, was, were, will, with*
- The general strategy for determining a stop list is to sort the terms by *collection frequency* (the total number of times each term appears in the document collection), and then to take the most frequent terms.

Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words
- But the trend is away from doing this:
  - Good compression techniques means the space for including stop words in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - You need them for:
    - Phrase queries: “King of Denmark”
    - Various song titles, etc.: “Let it be”, “To be or not to be”
    - “Relational” queries: “flights to London”
- Most web search engines index stop words.

Normalization to terms

- ***Token normalization*** is the process of choosing the most representative form of tokens so that matches occur despite superficial differences in the character sequences of the tokens.

- Result is terms: a **term** is a (normalized) word type, which is an entry in our IR system dictionary.

Normalization to terms

- Words in indexed text as well as query words must be **normalized** into the ***same form***
  - We want to match ***U.S.A.*** and ***USA***
- We most commonly ***implicitly*** define ***equivalence classes*** of terms by, e.g.,
  - deleting periods to form a term
    - ***U.S.A., USA*** (***USA)***
  - deleting hyphens to form a term
    - ***anti-discriminatory, antidiscriminatory*** (***antidiscriminatory)***

Case folding

Case folding refers to reducing all letters to lower case.

- Exception: upper case in mid-sentence?
  - e.g., ***General Motors***
  - ***Fed*** vs. ***fed***
- Often best to lower case everything, since users will use lowercase regardless of 'correct' capitalization.

- Google example:
  - Query ***C.A.T.***
    - #1 result is for "cat" (animal) *not* Caterpillar Inc.

Thesauri and soundex

- Do we handle synonyms and homonyms?
  - E.g., by hand-constructed equivalence classes
    - ***car*** = ***automobile*** ***color*** = ***colour***
  - We can rewrite to form equivalence-class terms
    - When the document contains ***automobile***, index it under ***car-automobile*** (and vice-versa)
  - Or we can expand a query
    - When the query contains ***automobile***, look under ***car*** as well
- What about spelling mistakes?
  - One approach is to use **Soundex** algorithm to form equivalence classes of words based on phonetic heuristics (***more on this later***).

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color.*
- Lemmatization implies doing "proper" reduction to dictionary headword form.

Stemming

- Reduce terms to their "roots" before indexing
- "Stemming" suggest crude affix chopping
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm

- Commonest algorithm for stemming English
  - Results suggest it's at least as good as other stemming options
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

- Porter's stemmer: http://www.tartarus.org/~martin/PorterStemmer/

Typical rules in Porter

Rule	Example
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat
ational → *ate*	
tional → *tion*	

Typical rules in Porter

- the *measure* of a word, which loosely checks the number of syllables to see whether a word is long enough that it is reasonable to regard the matching portion of a rule as a suffix rather than
- Weight of word sensitive rules
- *(m>1) EMENT →*
  - *replacement → replac*

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
  - http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  - Single-pass, longest suffix removal (about 250 rules)

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries,and decreases effectiveness for others.
- Porter Stemmer equivalence class *oper* contains all of *operate operating operates operation operative operatives operational*.
- Queries where stemming hurts: "operational AND research", "operating AND system", "operative AND dentistry"

- Do stemming and other normalizations help?
  - English: very mixed results. Helps recall for some queries but harms precision on others
    - E.g., operative (dentistry) ⇒ oper
  - Definitely useful for Spanish, German, Finnish, …
    - 30% performance gains for Finnish!

- Definitely useful for Spanish, German, Finnish, …30% performance gains for Finnish!

Exercise 1

Are the following statements true or false?

a. In a Boolean retrieval system, stemming never lowers precision.

b. In a Boolean retrieval system, stemming never lowers recall.

c. Stemming increases the size of the vocabulary.

d. Stemming should be invoked at indexing time but not while processing a query.

Exercise 2

Suggest what normalized form should be used for these words (including the word itself as a possibility):

a. ’Cos

b. Shi’ite

c. cont’d

d. Hawai’i

e. O’Rourke

Exercise 3

Applying Porter stemmer rule group,

what will the following words be stemmed to?

- *circus canaries boss*


Thank
You